

matchout

December 2, 2015
9:13

Contents

1 Read in two output files and compare contents	1
---	---

1 Read in two output files and compare contents

\$Id: e5ab5d0345890f5bcd07623c836fbe7931ff262 \$

Is used to compare the current DEGAS 2 output netCDF file (as named in `degas2.in`) with another from an equivalent run (all of the arrays must be exactly the same size). The path name to the latter file is specified on the input line. The most frequent use of `matchout` is to check that two separate runs have yielded results which are the same to within roundoff error. Because of the size of the ouptut netCDF file, determining this by visually comparing the files is impractical. The program can also be used to verify, at least in a cursory manner, that two runs are in statistical agreement.

The output is broken up into major sections according to the main output arrays:

output_grp(i) Contains the results from source group *i* without post-processing contributions.

output_all Is the sum of *output_grp* over source groups.

out_post_grp(i) Final results, including post-processing contributions, from source group *i*.

out_post_all Is the sum of *out_post_grp* over all source groups.

The reason for treating separately the arrays with post-processed data is that the impact of post-processing on the relative standard deviation is not computed. Hence, critical statistical comparisons must be made on results without the post-processing contributions. The comparisons *with* them are included to provide a secondary check and to verify the post-processing routines.

Each major section consists of a loop over all of the tallies in the problem. The output for an individual tally should look like:

```
ion momentum source vector :
  largest difference =      13427.334474026
  at indices ( 1919,  4,  1,  1,  1), component   3, itot =  122099
  largest relative difference =    679.45039474401
  at indices ( 1981,  6,  1,  1,  1), component   1, itot =  134601
  fraction within 1 sigma =    0.72405107270043
  "      "      2      "      0.93859115782159
  "      "      3      "      0.96855728307131
  fraction binned =    0.31154949396547
```

The first line gives the full name of the tally. The second line contains the largest difference, for that tally, between the values contained in the current output netCDF file and those in the file specified on the command line. The difference is normalized to the value in the file specified on the input line (the hope is that it does not matter which of the two values is used for normalization).

The next line provides information about the independent variables associated with the largest difference. First, the indices of the independent variables are given. These will range from 1 to the value of *tally_tab_index*. If the tally is a vector, the “component” (1, 2, or 3) is specified. Finally, *itot* is the corresponding index in the full array (e.g., *output_all_itot,o_mean*).

The next two lines do the same for the largest relative difference. The “relative” means relative to an effective standard deviation,

$$\sigma_{\text{eff}} = (\sigma_1^2 + \sigma_2^2)^{1/2}. \quad (1)$$

The relative differences are binned to allow an overall estimate of the error. Relative differences less than one are counted in the “fraction within 1 sigma” bin and so on. All values for the tally are counted equally, except that trivial ones (e.g., both are zero) are ignored. The fraction of all (including trivial ones) of the data values that are binned (including those with relative differences greater than three) is given in the last line. If all of the binned data values were independent (in reality they are not), and the two sets of results were in statistical agreement, the binning fractions would be near their ideal values, 68%, 95%, and 99.7%. The “fraction binned” will alert the user to situations where too few values were binned to allow a statistical comparison.

In comparing runs which should be numerically identical, the largest differences (absolute and relative) should be close to machine precision, say, between 10^{-13} and 10^{-15} . If the run is carried out on the same platform for both files, differences of zero usually result. Accumulated or accentuated roundoff error can legitimately lead to differences as large as 10^{-10} for a single tally or data point (the neutral flux vector tally typically results in the greatest differences). Larger or more persistent differences should be investigated.

```
"matchout.f" 1 ==
@m FILE 'matchout.web'
```

The main program

```
"matchout.f" 1.1 ≡
program matchout
  implicit none_f77
  implicit none_f90
  character*FILELEN file2
  call readfilenames
  @#if 0
    call read_geometry
    call nc_read_species
    call nc_read_reactions
    call nc_read_materials
    call nc_read_pmi
    call nc_read_problem
    call nc_read_background
    call nc_read_tally
    call nc_read_output
  @#endif
  call degas_init
  call nc_read_output
  call command_arg(1, file2)
  call match_data(file2)
  stop
end
```

⟨ Functions and Subroutines 1.2 ⟩

Compare data in two output files. Having read in one, copy its contents to local arrays, read in the data from the second file, and then do the required comparisons.

```

⟨ Functions and Subroutines 1.2 ⟩ ≡
subroutine match_data(file2)
  define_varp(one_all, FLOAT, output_moments.ind, output_tab.ind)
  define_varp(one_grp, FLOAT, output_moments.ind, output_tab.ind, output_grps.ind)
  define_varp(one_post_all, FLOAT, output_moments.ind, output_tab.ind)
  define_varp(one_post_grp, FLOAT, output_moments.ind, output_tab.ind, output_grps.ind)
  implicit none f77
  pr_common
  so_common
  tl_common
  ou_common
  rf_common
  implicit none f90

  integer i, is
  real sig_max
  character*FILELEN file2

  ⟨ Memory allocation interface 0 ⟩

  declare_varp(one_all)
  declare_varp(one_grp)
  declare_varp(one_post_all)
  declare_varp(one_post_grp)

  var_alloc(one_all)
  var_alloc(one_grp)
  var_alloc(one_post_all)
  var_alloc(one_post_grp)

  assert(output_old_file ≡ TRUE)
  do i = 0, tally_size - 1
    one_alli,o_mean = output_alli,o_mean
    one_alli,o_var = output_alli,o_var
    one_post_alli,o_mean = out_post_alli,o_mean
    one_post_alli,o_var = output_alli,o_var // May change this!
  end do

  do is = 1, so_grps
    do i = 0, tally_size - 1
      one_grpis,i,o_mean = output_grpis,i,o_mean
      @#if 0
        one_grpis,i,o_var = output_grpis,i,o_var
      @#else
        one_grpis,i,o_var = out_post_grpis,i,o_var
      @#endif
      one_post_grpis,i,o_mean = out_post_grpis,i,o_mean
      @#if 0
        one_post_grpis,i,o_var = output_grpis,i,o_var // May change!
      @#else
        one_post_grpis,i,o_var = out_post_grpis,i,o_var
      @#endif

```

```

    end do
end do

/* Explicitly free these since they will all be reallocated when the second file is read in (although
   the arrays had better be the same size!). */
var_free(output_all)
var_free(output_grp)
var_free(out_post_all)
var_free(out_post_grp)
var_free(output_2D_coupling)
var_free(output_weight_grp)
var_free(output_num_flights)
var_free(output_random_seed)

filenames_array outputfile = file2
call nc_read_output

@if 0
// Now doing this for the output file itself.
do i = 0, tally_size
  out_post_all_i,o_var = output_all_i,o_var // May change
end do

do is = 1, so_grps
  do i = 0, tally_size - 1
    out_post_grp_is,i,o_var = output_grp_is,i,o_var // May change
  end do
end do

@else
do is = 1, so_grps
  do i = 0, tally_size - 1
    output_grp_is,i,o_var = out_post_grp_is,i,o_var
  end do
end do
@endif

sig_max = const(1., 2)

do is = 1, so_grps
  write(stdout, *) 'output_grp(', is, ')'
  call compare_output(output_grp_is,0,o_mean, one_grp_is,0,o_mean, sig_max)
end do

write(stdout, *) 'output_all'
call compare_output(output_all_0,o_mean, one_all_0,o_mean, sig_max)

write(stdout, *) 'NOTE: Post-processing variances may be incorrect'
do is = 1, so_grps
  write(stdout, *) 'out_post_grp(', is, ')'
  call compare_output(out_post_grp_is,0,o_mean, one_post_grp_is,0,o_mean, sig_max)
end do

write(stdout, *) 'out_post_all'
call compare_output(out_post_all_0,o_mean, one_post_all_0,o_mean, sig_max)

/* Should call clear_output as well, but needs to be moved to some file other than flighttest.web
   to avoid make conflicts. */

```

```
var_free(one_all)
var_free(one_grp)
var_free(one_post_all)
var_free(one_post_grp)

return
end
```

See also section 1.3.

This code is used in section 1.1.

Compare two individual output arrays, tally-by-tally.

```

⟨ Functions and Subroutines 1.2 ⟩ +≡
subroutine compare_output(scores1, scores2, sig_max)
  implicit none f77
  pr_common
  tl_common // Common
  implicit none f90
  real sig_max // Input
  real scores10:*,o_mean:o_var, scores20:*,o_mean:o_var
  integer itot, jscore, ic_max, i_max, j_max, k_max, /* Local */
           l_max, m_max, itot_max, ic_rel_max, i_rel_max, j_rel_max, k_rel_max, l_rel_max, m_rel_max,
           itot_rel_max, bin1, bin2, bin3, binelse, bin_tot, i, j, k, l, m, ic
  integer ind_val5
  real x1, x2, sig1, sig2, sig_eff, diff, diff_rel, delta_max, delta_rel_max
  ⟨ Memory allocation interface 0 ⟩
  st_decls

itot = -1
do jscore = 1, tl_num
  delta_max = -one
  delta_rel_max = -one
  ic_max = -1
  i_max = -1
  j_max = -1
  k_max = -1
  l_max = -1
  m_max = -1
  itot_max = -1
  ic_rel_max = -1
  i_rel_max = -1
  j_rel_max = -1
  k_rel_max = -1
  l_rel_max = -1
  m_rel_max = -1
  itot_rel_max = -1
  bin1 = 0
  bin2 = 0
  bin3 = 0
  binelse = 0
do m = 1, tally_tab_indexjscore,5
  ind_val5 = m - 1
do l = 1, tally_tab_indexjscore,4
  ind_val4 = l - 1
do k = 1, tally_tab_indexjscore,3
  ind_val3 = k - 1
do j = 1, tally_tab_indexjscore,2
  ind_val2 = j - 1
do i = 1, tally_tab_indexjscore,1
  ind_val1 = i - 1
do ic = 1, tally_dep_var_dimjscore
  itot ++

```

```

x1 = scores1_out_array_index(ic, ind_val, jscore),o_mean
x2 = scores2_out_array_index(ic, ind_val, jscore),o_mean
sig1 = scores1_out_array_index(ic, ind_val, jscore),o_var
sig2 = scores2_out_array_index(ic, ind_val, jscore),o_var
sig_eff = sqrt(sig12 + sig22)
if (x1 ≠ zero ∧ x2 ≠ zero) then
    diff = abs(x1 - x2) / abs(x1)
else if (x1 ≡ zero ∧ x2 ≡ zero) then
    diff = zero
else
    diff = one      // One's = 0, the other's not!
end if

assert(sig_eff ≥ zero)
if (sig_eff < sig_max ∧ sig_eff ≠ zero) then
    diff_rel = diff / sig_eff
    if (diff_rel < one) then
        bin1 ++
    else if (diff_rel < two) then
        bin2 ++
    else if (diff_rel < const(3.)) then
        bin3 ++
    else
        binelse ++
    end if
else
    diff_rel = zero
end if
if (diff > delta_max) then
    ic_max = ic
    i_max = i
    j_max = j
    k_max = k
    l_max = l
    m_max = m
    itot_max = itot
    delta_max = diff
endif

if (diff_rel > delta_rel_max) then
    ic_rel_max = ic
    i_rel_max = i
    j_rel_max = j
    k_rel_max = k
    l_rel_max = l
    m_rel_max = m
    itot_rel_max = itot
    delta_rel_max = diff_rel
end if
end do
end do
end do
end do

```

```

    end do
end do
write (stdout, *) 'uuu', trim(tally_name_jscore), 'u:'
write (stdout, *) 'uuuuuuu'largest_difference' =', delta_max
write (stdout, *) 'uuuuuuuuu'at_indices(', i_max, ', ', j_max, ', ', k_max, ', ', l_max, ', ',
    m_max, ')', component', ic_max, ', ', itot=', itot_max
write (stdout, *) 'uuuuuuu'largest_relative_difference' =', delta_rel_max
write (stdout, *) 'uuuuuuuuu'at_indices(', i_rel_max, ', ', j_rel_max, ', ', k_rel_max, ', ',
    l_rel_max, ', ', m_rel_max, ')', component', ic_rel_max, ', ', itot=', itot_rel_max
bin_tot = bin1 + bin2 + bin3 + binelse
if (bin_tot > zero) then
    write (stdout, *) 'uuuuuuu'fraction_within_1_sigma' =', areal(bin1) / areal(bin_tot)
    write (stdout, *) 'uuuuuuuuu'uuuuuuu"uuuuuuu"uuuu2uuu"uuuuu', areal(bin1 + bin2) / areal(bin_tot)
    write (stdout,
        *) 'uuuuuuuuu"uuuuuuu"uuuu3uuu"uuuuu', areal(bin1 + bin2 + bin3) / areal(bin_tot)
end if
write (stdout, *) 'uuuuuuu'fraction_binned' =', areal(bin_tot) / (tally_tab_index_jscore,1 *
    tally_tab_index_jscore,2 * tally_tab_index_jscore,3 * tally_tab_index_jscore,4 * tally_tab_index_jscore,5 *
    tally_dep_var_dim_jscore)
end do
return
end

```

abs: 1.3.
areal: 1.3.
assert: 1.2, 1.3.

bin_tot: 1.3.
binelse: 1.3.
bin1: 1.3.
bin2: 1.3.
bin3: 1.3.

clear_output: 1.2.
command_arg: 1.1.
compare_output: 1.2, 1.3.
const: 1.2, 1.3.

declare_varp: 1.2.
define_varp: 1.2.
degas_init: 1.1.
delta_max: 1.3.
delta_rel_max: 1.3.
diff: 1.3.
diff_rel: 1.3.

FILE: 1.
FILELEN: 1.1, 1.2.
filenames_array: 1.2.
file2: 1.1, 1.2.
flighttest: 1.2.
FLOAT: 1.2.
i: 1.2, 1.3.

i_max: 1.3.
i_rel_max: 1.3.
ic: 1.3.
ic_max: 1.3.
ic_rel_max: 1.3.
implicit_none_f77: 1.1, 1.2, 1.3.
implicit_none_f90: 1.1, 1.2, 1.3.
ind_val: 1.3.
is: 1.2.
itot: 1, 1.3.
itot_max: 1.3.
itot_rel_max: 1.3.

j: 1.3.
j_max: 1.3.
j_rel_max: 1.3.
jscore: 1.3.

k: 1.3.
k_max: 1.3.
k_rel_max: 1.3.

l: 1.3.
l_max: 1.3.
l_rel_max: 1.3.

m: 1.3.
m_max: 1.3.
m_rel_max: 1.3.
match_data: 1.1, 1.2.
matchout: 1.1.

nc_read_background: 1.1.
nc_read_materials: 1.1.
nc_read_output: 1.1, 1.2.
nc_read_pmi: 1.1.
nc_read_problem: 1.1.
nc_read_reactions: 1.1.
nc_read_species: 1.1.
nc_read_tally: 1.1.

o_mean: 1, 1.2, 1.3.
o_var: 1.2, 1.3.
one: 1.3.
one_all: 1.2.
one_grp: 1.2.
one_post_all: 1.2.
one_post_grp: 1.2.
ou_common: 1.2.
out_array_index: 1.3.
out_post_all: 1.2.
out_post_grp: 1, 1.2.
output_all: 1, 1.2.
output_grp: 1, 1.2.
output_grps_ind: 1.2.

output_moments_ind: 1.2.
output_num_flights: 1.2.
output_old_file: 1.2.
output_random_seed: 1.2.
output_tab_ind: 1.2.
output_weight_grp: 1.2.
output_2D_coupling: 1.2.
outputfile: 1.2.

pr_common: 1.2, 1.3.

read_geometry: 1.1.
readfilenames: 1.1.
rf_common: 1.2.

scores1: 1.3.
scores2: 1.3.
sig_eff: 1.3.
sig_max: 1.2, 1.3.
sig1: 1.3.
sig2: 1.3.
so_common: 1.2.
so_grps: 1.2.
sqrt: 1.3.
st_decls: 1.3.
stdout: 1.2, 1.3.

tally_dep_var_dim: 1.3.
tally_name: 1.3.
tally_size: 1.2.
tally_tab_index: 1, 1.3.
tl_common: 1.2, 1.3.
tl_num: 1.3.
trim: 1.3.
TRUE: 1.2.
two: 1.3.

var_alloc: 1.2.
var_free: 1.2.

web: 1.2.

x1: 1.3.
x2: 1.3.
zero: 1.3.

⟨ Functions and Subroutines 1.2, 1.3 ⟩ Used in section 1.1.
⟨ Memory allocation interface 0 ⟩ Used in sections 1.3 and 1.2.

COMMAND LINE: "fweave -f -i! -W[-ykw800 -ytw40000 -j -n/
/Users/dstotler/degas2/src/matchout.web".

WEB FILE: "/Users/dstotler/degas2/src/matchout.web".

CHANGE FILE: (none).

GLOBAL LANGUAGE: FORTRAN.